ภาษา LOG Ø : 1

เกริ่นนำ

บุญเลิศ เอี่ยมทัศนา

อีกครั้งหนึ่งครับที่ผมนำภาษาชื่อแปลกมาเอ่ยถึง ภาษานี้อ่านเป็นคำเต็มว่า "ภาษา ล็อก-ซี-โร" แต่ผมจะขอใช้คำว่า "ภาษาล็อกศูนย์" แทน และเช่นเดียวกับที่เคยกล่าวถึงภาษาไอดิวซีโรคือ จะเป็นภาษาเก่าก็ไม่ใช่ใหม่ก็ไม่เชิง เพราะผมนำมาจากภาษาโลโก

แต่ความมุ่งหมายในการนำเสนอต่างกันครับคือ ครั้งนี้ผมต้องการ (ให้ช่วยกันพัฒนา) ให้เป็นภาษาที่ใช้งานได้จริง ๆ สำหรับเยาวชนของเรา ส่วนผลพลอยได้คือ แสดงถึงการเขียนโปรแกรมตัวแปลภาษาในแบบอินเตอร์เพรตเตอร์ (แบบง่าย ๆ) ซึ่งคุณสามารถนำไปประยุกต์ใช้ในการเขียนโปรแกรมในยุคที่สามได้

TKS กับ LOGO และ PROLOG

ในกระบวนภาษาคอมพิวเตอร์ทั้งหลาย ภาษาที่ผมเห็นว่าควรเขียนโปรแกรมตัวแปลภาษา เพื่อให้ผู้ใช้ (โปรแกรมเมอร์) เขียนโปรแกรมในภาษานั้น ๆ เป็นภาษาไทยได้คือ ภาษาโลโกกับภาษาโปรล็อก ทั้งนี้เพราะผู้เขียนโปรแกรมในภาษาทั้งสองนี้เป็นหรือเกือบเป็นผู้ใช้คนสุดท้าย (end user) ของภาษาทั้งสองนี้แล้ว เช่น ในภาษาโปรล็อกแทนที่จะเขียนโปรแกรมว่า

father (dum, dumrong).

จะเขียนได้เป็น

พ่อ (ดำ, ดำรงค์).

สำหรับภาษาโลโกดังแสดงในรูปที่ 1

ที่ผมไม่นำมากล่าวถึงเพราะ หนึ่งอยู่ในระยะค้นคว้า สอง นับตั้งแต่มีรหัสอักขระมาตรฐานสมอ.ออกมา ผมรู้สึกอึดอัดใจมากเพราะไม่เป็นหนึ่งเดียว กลายเป็นสมอ.ของ ฯลฯ ซึ่งไม่เป็นมาตรฐานเดียวกัน นอกจากจะทำเป็นภาษาไทย 4 ระดับ

ดังนั้นหากผมต้องการใช้ภาษาไทย 2 หรือ 3 ระดับ ผมต้องเลือกการ์ดหนึ่งการ์ดใดมาอ้างถึง ทำให้ใช้ไม่ได้ทั่วไป เมื่อทางมหาวิทยาเกษตรศาสตร์มีโครงการทำภาษาไทยระดับแก่นคือ TKS (ไมโครฯ 45) ซึ่งเป็นภาษาไทยในแบบกราฟิกไม่ขึ้นกับคาแร็กเตอร์เยนฯ คือไม่ขึ้นกับการ์ด หากการ์ดนั้นสามารถแสดงภาพในแบบกราฟิกได้เป็นใช้ได้

นอกจากนี้ทางมหาวิทยาลัยเกษตรศาสตร์ยังจัดทำซอฟต์แวร์ทางด้านกราฟิกอีกด้วย จึงเหมาะที่ผมจะนำมาใช้ในการเขียนโปรแกรมตัวแปลภาษาโลโกเป็นอย่างยิ่ง

ก่อนที่จะผิดหวังหรือสมหวัง

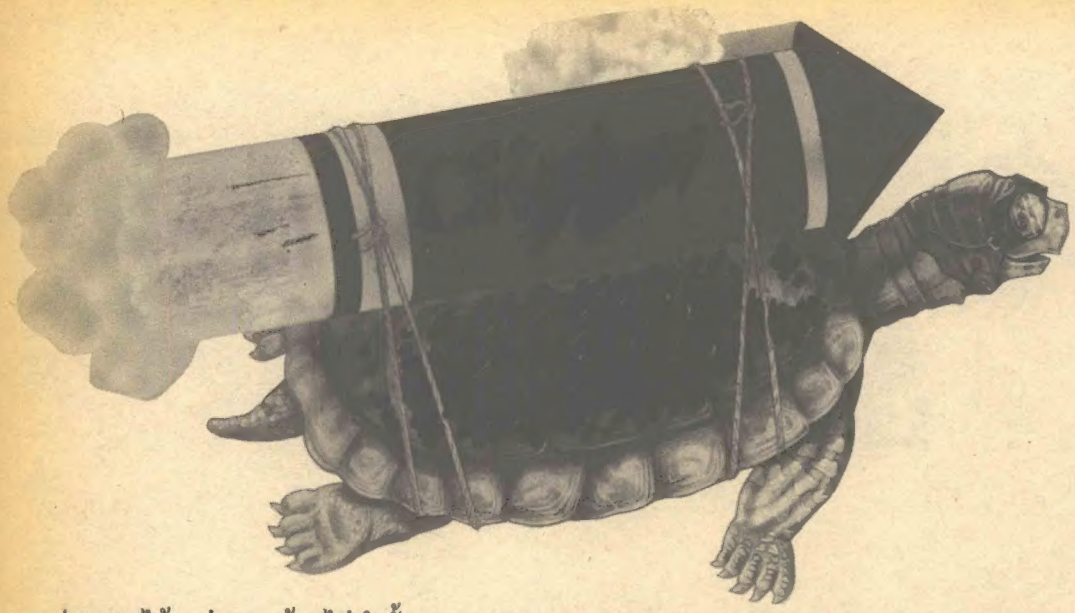
ข้อดีของภาษาไทยระดับแก่นของมหาวิทยาลัยเกษตรศาสตร์อีกประเด็นหนึ่งที่ผมนำมาพิจารณาคือ เมื่อดำเนินการเสร็จแล้วจะเป็นโปรแกรมที่ฝังตัวอยู่ในหน่วยความจำ ซึ่งเราสามารถเรียกใช้ได้ในแบบฟังก์ชันคอลล์ของดอส

แต่ในขณะนี้ยังอยู่ในขั้นตอนของการพัฒนาจึงเป็นเพียงโปรแกรมย่อยสำหรับภาษาซี ส่วนผมต้องการเขียนโปรแกรมตัวแปลภาษาด้วยภาษาปาสคาล จึงต้องรอ หรือไม่ก็ดำเนินการให้ใช้กับ

```
ทำ กังหัน :มุม
 สี่เหลี่ยม 50
 หันขวา :มุม
 กังหัน :มุม
จบ
```

```
TO SPINSQUARE :ANGLE
  SQUARE 50
  RIGHT :ANGLE
  SPINSQUARE :ANGLE
END
```

รูปที่ 1 แสดงโปรแกรมภาษาโลโก ภาษาอังกฤษ เทียบกับภาษาไทย

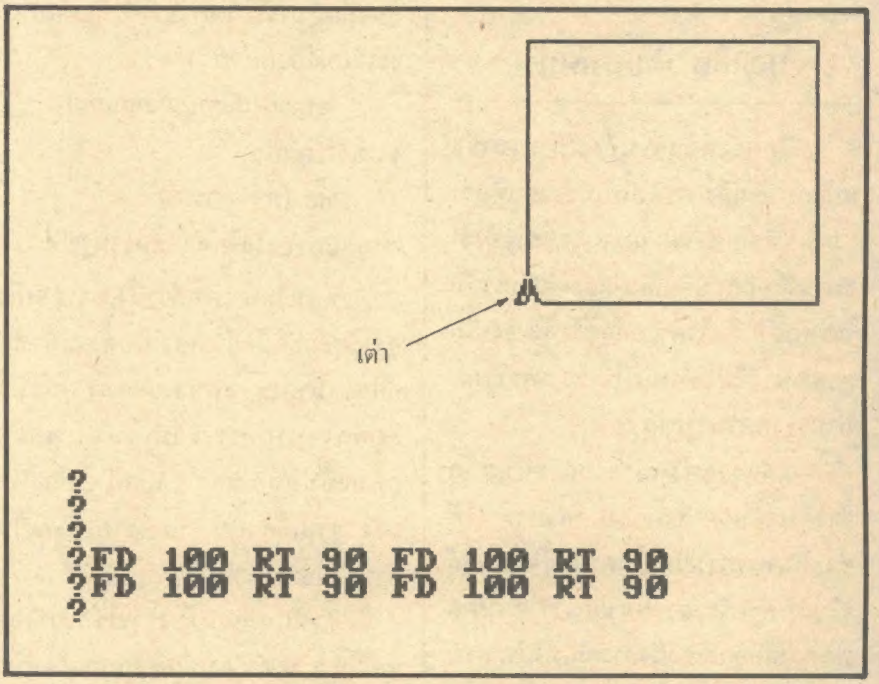


ปาสคาลได้ แต่ผมจะต้องไม่ทำทั้งสองอย่างครับ

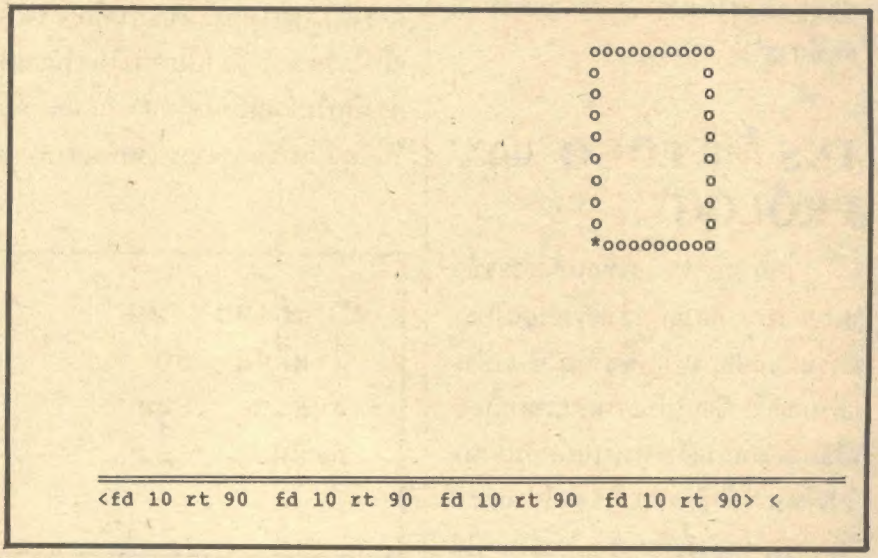
เมื่อเป็นเช่นนี้ผมจะต้องลดระดับมาเป็นเพียงล็อกศูนย์คือ ต้องเขียนโปรแกรมในภาษาล็อกศูนย์เป็นภาษาอังกฤษไปก่อน และที่เป็นล็อกศูนย์ก็เพราะว่าในขั้นแรกนี้ผมจะยังไม่ติดตั้งกราฟิกจะเป็นเพียงแมงมุมคือเครื่องหมายดอกจันเดินไปเดินมาบนจอภาพแทนเต่าก่อน ทั้งนี้เพื่อจะเน้นถึงการเขียนอินเตอร์เพรตเตอร์

เมื่อจบจากอินเตอร์เพรตเตอร์แล้วผมจะนำกราฟิกมาติดตั้ง ก็จะเป็นภาษาโลโกโดยสมบูรณ์ (คิดไว้ว่า) ก็จะเป็นข้อดีอีกประการหนึ่งเพราะใช้กับการ์ดเฮอร์คิวลิสโดยตรง (ไอบีเอ็มโลโกต้องใช้กับการ์ดสี) และก็ตัดปัญหาที่ผมห่วงอยู่คือ การให้เด็กคุ้นเคยกับคีย์อักษรไทยน่าจะทำได้ยากกว่าคีย์อักษรอังกฤษ โดยสรุปเราจะมี 2 เวอร์ชันครับ คือเวอร์ชันไทยและเวอร์ชันอังกฤษ

อีกเรื่องหนึ่งผมเขียนเรื่องนี้ต่างกว่าเคยคือ เท่าที่ผ่านมาผมจะเขียนโปรแกรมจนเสร็จสมบูรณ์แล้วจึงนำมาเขียนเรื่อง แต่บทความนี้ผมจะดำเนินการไปพร้อม ๆ กันเพราะเป็นเรื่องยาว ต้องการเขียนในแต่ละเรื่องย่อยสด ๆ ทันทีที่คิดได้ จึงอาจมีปัญหาอยู่ข้างหน้าหากเดินผิดทางก็ต้องย้อนกลับมาแก้โปรแกรมกันบ้างครับ



รูปที่ 2 แสดงคำสั่งให้วาดรูปสี่เหลี่ยมและภาพที่ปรากฏบนจอภาพของโลโก



รูปที่ 3 แสดงคำสั่งให้วาดรูปสี่เหลี่ยมและภาพที่ปรากฏบนจอภาพของล็อกศูนย์

เพื่อให้เป็นการเปรียบเทียบผมได้แสดงภาพบนจอภาพเมื่อใช้โลโก (ตัวจริง) ในรูปที่ 2 และเมื่อใช้ล็อกศูนย์ในรูปที่ 3 ทั้งคู่นี้ให้วาดรูปสี่เหลี่ยมเหมือนกัน แต่จะเห็นว่าแตกต่างกันมาก แต่เมื่อโครงการนี้เสร็จสมบูรณ์แล้วจะเหมือนกันหรือเกือบเหมือนกัน ซึ่งผมไม่ทราบว่าจะใช้ชื่อภาษาว่าอะไร (อาจเป็น ThaiLogo)

ภาษาโลโก

เนื่องจากเราไม่ค่อยกล่าวถึงภาษาโลโกกันนักผมจึงขอนำมากล่าวถึงโดยย่อครับ กล่าวคือ ศจ.เซมู เปเปร์ (Seymour Papert) แห่ง MIS (Massachusetts Institute of Technology) เป็นผู้สร้างภาษาโลโกขึ้นเมื่อประมาณ พ.ศ. 2513

ด้วยจุดประสงค์ที่จะให้เด็กตั้งแต่รู้ความสามารถใช้คอมพิวเตอร์ได้ ในแบบ "การเรียนรู้โดยไม่ต้องสอน" (learning without teaching) คือให้เด็กสามารถใช้โลโกได้ด้วยการเรียนรู้ด้วยตนเองประกอบคำแนะนำเพียงเล็กน้อย ไม่ใช่การเรียนแบบในชั้นเรียน (ถึงแม้จะจัดชั้นเรียนก็ให้เรียนในแบบเพลิดเพลิน) โดยทั่วไปจึงถือว่าภาษาโลโกไม่ใช่ภาษาใช้งาน แต่ก็สามารถนำไปใช้งานทางด้านปัญญาประดิษฐ์ได้ในบางเรื่อง

โดยที่เราสามารถแบ่งการใช้งานภาษาโลโกออกได้เป็น 3 แขนงคือ

หุ่นยนต์ หรืออาจจะมองเป็นงานทางด้านจิตรกรก็ได้คือ ในภาษาโลโกจะมีคำสั่งชุดหนึ่งเป็นคำสั่งที่จะสั่งให้เต่า (ทำเป็นรูปสามเหลี่ยมเล็ก ๆ) เคลื่อนไหวไปมาบนจอภาพซึ่งจะให้ทิ้งรอยทางเดินไว้บนจอภาพหรือไม่ก็ได้ ซึ่งถ้าดูในลักษณะของหุ่นยนต์ก็เป็นการฝึกให้เด็กบังคับหุ่นยนต์ แต่ถ้าดูในแง่ของการวาดภาพจะเห็นว่าตัวเต่าคือปากกาและรอยทางเดินคือเส้นปากกาเทียบได้กับการวาดภาพบนจอภาพ

ภาษาศาสตร์ ในภาษาโลโกมีข้อมูบแบบลิสต์ (List) ทำให้สามารถเขียนโปรแกรมทางด้านภาษาธรรมชาติ (Natural language) เช่น การตรวจสอบประโยคในภาษาอังกฤษว่าถูกไวยากรณ์หรือไม่ ได้เช่นเดียวกับภาษาลิสป์ (LISP) และโปรล็อก (PROLOG)

คณิตศาสตร์ นอกจากนี้ยังมีฟังก์ชั่นทางคณิตศาสตร์ เช่นเดียวกันกับที่มีในภาษาปาสคาลและซี เพื่อใช้ในการคำนวณต่าง ๆ โดยเฉพาะทางด้านตรีโกณมิติ

จะเห็นว่าในภาษาโลโกเพียงภาษาเดียวประกอบด้วยภาษาคอมพิวเตอร์ถึง 3 แขนงคือ ตัวของมันเอง ลิส์ป-โปรล็อก และปาสคาล-ซี สนองตอบความต้องการของผู้ใช้ได้ตั้งแต่วัยเด็ก (เล็ก) จนถึงก่อนเข้ามหาวิทยาลัย แต่ที่รู้จักกันมากที่สุดคือเรื่องของเต่า แทบจะกล่าวได้ว่าเมื่อกล่าวถึงภาษาโลโกจะหมายถึงเต่า ซึ่งในโครงงานนี้ผมก็จะเน้นหนักถึงเรื่องเต่าเท่านั้นครับ

เด็กกับโลโก

จากที่กล่าวมาจะเห็นว่าภาษาโลโกเหมาะสำหรับเด็กเล็กไปจนถึงผู้ใหญ่ และสำหรับเรื่องของเต่าแล้ว เหมาะอย่างยิ่งสำหรับเด็กในวัยซน ซึ่งผมก็ต้องขอเรียนทำความเข้าใจก่อนว่า การที่เราให้เด็กใช้ภาษาโลโกนั้นไม่ได้มุ่งหมายให้เด็กหัดเขียนโปรแกรมภาษาคอมพิวเตอร์

หากมุ่งหมายที่จะให้เด็กเกิดความคิดริเริ่ม มีความคิดสร้างสรรค์ ฝึกทำงานอย่างเป็นระเบียบมีขั้นตอนในการทำงาน ส่วนผลพลอยได้คือ ทำให้เด็กได้ใกล้ชิดกับเครื่องคอมพิวเตอร์ คุ้นเคยกับคีย์บอร์ด (ซึ่งจำเป็นสำหรับนักบริหารในอนาคต และความจริงคือจำเป็นสำหรับทุกสาขาอาชีพ หากจะเป็น NIC กัน) และท้ายสุดคือรู้จักการเขียนโปรแกรมในแบบเป็นโครงสร้าง

ด้วยเหตุนี้ผมจึงขวนขวายที่จะให้เขียนโปรแกรมได้ด้วยภาษาไทย เช่น เด็กต้องการให้เต่าวาดรูปดอกพุดซ้อน เมื่อต้องเขียนเป็นภาษาอังกฤษจะต้องเสียเวลาในการคิดคำภาษาอังกฤษ ซึ่งอาจมากกว่าเวลาที่ใช้ในการคิดเขียนโปรแกรมสั่งเต่า และทำให้เสียจินตนาการด้วย

ลักษณะของภาษาโลโก

เมื่อเรารัน LOGO คือโปรแกรมตัวแปลภาษาโลโกแล้วขอกล่าวอย่างรวบรัดว่าจอภาพจะถูกแบ่งออกเป็น 2 ตอน ส่วนบนประมาณกึ่งกลางจอภาพมีรูปเต่า (เป็นรูปสามเหลี่ยม) คอยอยู่สำหรับส่วนล่างจะแสดงเคอร์เซอร์เพื่อให้เราป้อนคำสั่ง คำสั่งแบ่งออกได้เป็น 3 ลักษณะคือ

คำสั่งเครื่อง

เป็นคำสั่งที่ต้องการให้เครื่องดำเนินการให้เราซึ่งส่วนมากได้แก่ คำสั่งที่เกี่ยวกับไฟล์ เช่น load, save และ list ซึ่งไฟล์นี้หมายถึงไฟล์ในภาษาโลโก สำหรับ list เป็นการให้แสดงโปรแกรมในภาษาโลโกที่เก็บอยู่ในหน่วยความจำ (โหลดลงมาหรือป้อนด้วยเอดิเตอร์)

คำสั่งเต่า

คือคำสั่งที่ต้องการให้เต่าดำเนินการ เช่น FD 20 RT 90 FD 20 เป็นการสั่งให้เต่าเดินไปข้างหน้า 20 ก้าว หันขวา 90 องศา แล้วเดินไปข้างหน้าอีก 20 ก้าว ซึ่งเมื่อเต่าเดินจะทิ้งรอยเท้าเกิดเป็นภาพบนจอภาพ ดังแสดงในรูปที่ 2

เรียกใช้เอดิเตอร์

เมื่อป้อนคำว่า TO ตามด้วยชื่อโปรแกรม (ย่อย) เช่น TO BOX หลังจากกดคีย์ Enter แล้วโลโกจะให้เรา

ป้อนโปรแกรมได้โดยใช้เอดิเตอร์ในตัวในแบบ One line editor แต่ถ้าใช้คำสั่ง EDIT ตามด้วยชื่อโปรแกรมย่อย จะเป็นการเรียกใช้เอดิเตอร์อีกตัวหนึ่ง ในแบบ Full screen editor

การสอนโลโก

ผมคิดว่าหากนำเรื่องราวการสอนโลโกแก่เด็กมากล่าวถึง จะช่วยให้ท่านที่ไม่คุ้นเคยกับภาษาโลโกได้ติดตามบทความของผมต่อไปได้โดยสะดวก จึงขอนำมากล่าวถึงก่อนคือ ในการสอนโลโกแก่เด็กส่วนมากแล้วจะดำเนินการดังนี้

- เริ่มแรกจะให้เด็กหัดใช้คำสั่งเต่าโดยตรงจากคีย์บอร์ด ให้เต่าเดินไปเดินมา

- เมื่อสังเกตเห็นว่าเด็กเริ่มเบื่อที่ต้องใช้คำสั่งเดิมซ้ำแล้วซ้ำอีก ก็จะเริ่มสอนให้เขียนโปรแกรมเท่ากับเป็นการสอนให้ใช้เอดิเตอร์ไปในตัว

- ในขั้นตอนนี้เอง หรือเมื่อสังเกตเห็นว่าเด็กเริ่มเบื่อที่ต้องป้อนโปรแกรมทุกครั้งที่ต้องการใช้ ก็จะสอนให้รู้จักคำสั่ง save, load และ list

- เมื่อเด็กเริ่มเขียนโปรแกรมยาวขึ้นและชักเวียนหัวที่ต้อนเขียนโปรแกรมยาว ๆ ก็จะสอนให้รู้จักใช้วงรอบ repeat และโปรแกรมเรียกตนเองดังแสดงในรูปที่ 4

- จากนี้ก็เป็นเรื่องของเด็กโตที่จะคิดเขียนโปรแกรมทางด้านภาษาศาสตร์และการคำนวณต่อไป

ดังนั้นจุดมุ่งหมายของผมในโครงงานนี้จึงจะเป็นเรื่องของเต่าโดยเฉพาะ แต่จะเขียนโปรแกรมเปิดไว้ให้สำหรับภาษาศาสตร์ และการคำนวณต่อไปด้วย

วางแนวความคิด

จากเรื่องราวที่กล่าวมาจะเห็นว่าโครงงานนี้แบ่งออกได้เป็น 3 โครงงานย่อยคือ คำสั่งเครื่อง คำสั่งเต่า และเอดิเตอร์ ซึ่งทั้งสามโครงงานย่อยนี้อาจเขียนคนเดียวต่อเนื่องกันไปหรือแยกกันเขียนก็ได้ กล่าวคือ (คำภาษาอังกฤษต่อไปนี้เป็นคำที่ตั้งขึ้นเพื่อใช้ในโปรแกรม)

```
TO  CIRCLE
  REPEAT 360 [FD 1  RT 1]
END
```

รูปที่ 4 โปรแกรมสร้างวงกลมโดยใช้วงรอบ REPEAT

โครงงานคำสั่งเครื่อง

คือ ConsoleCommand เป็นโครงงานพื้นฐานธรรมดา

โครงงานคำสั่งเต่า

ใช้ชื่อว่า TurtleCommand ซึ่งคำสั่งเต่าแยกออกได้เป็น 2 แบบคือ คำมาตรฐาน (StandardIdentifier) ซึ่งเราผู้เขียนโปรแกรมเป็นผู้กำหนด กับคำที่ผู้ใช้กำหนด (UserDefinded) และสามารถแยกโครงงานนี้ได้เป็น 2 โครงงานย่อย คือ

- Interpreter เพื่อแปลคำที่ผู้ใช้กำหนดให้เต่าสามารถปฏิบัติได้ ก็คือการเขียนอินเตเพรตเตอร์นั่นเอง (จึงใช้ชื่อนี้) เป็นโครงงานที่ค่อนข้างยาก

- Library คือคำมาตรฐาน ผมคิดตั้งชื่อโครงงานนี้เช่นนี้ เพราะต้องการแยกออกไปติดตั้งภายหลังดังกล่าวในตอนต้น เป็นโครงงานที่ไม่ยากแต่ต้องการความรู้ทางด้านกราฟิก

โครงงานเอดิเตอร์

ใช้ชื่อว่า Editor เป็นโครงงานที่ไม่ยากแต่ยุ่ง เพราะมีรายละเอียดปลีกย่อยมาก ผมจึงเก็บไว้เขียนภายหลังและตั้งใจจะให้เป็นเอดิเตอร์ขนาดเล็กที่ใช้งานได้ทั่วไป โดยที่โปรแกรมที่ต้องการใช้เอดิเตอร์เพียงแต่ส่งแอดเดรสกับขนาดของบัฟเฟอร์ และตำแหน่งสุดท้ายของข้อมูลเดิมในบัฟเฟอร์ไปให้เท่านั้น ในระหว่างนี้จึงต้องใช้เอดิเตอร์ของเทอร์โบปาสคาลไปก่อนครับ

ยูนิต LOG00

โครงงานย่อยทั้งสามดังกล่าวมานี้ ผมจะแยกเขียนเป็นยูนิต (เพื่อแสดงว่าหากแยกกันเขียนจะเขียนได้อย่างไร) และอาจจะแยกเป็นยูนิตรอง ๆ ต่อไปอีก แต่มีเรื่องจำนวนหนึ่งซึ่งใช้ร่วมกันและต้องเขียนขึ้นก่อน ผมจึงจัดทำเป็นยูนิตแรกให้ชื่อว่ายูนิต LOG00 ดังแสดงในรูป A1 ถึง A10

หากคุณสงสัยว่าผมทราบได้อย่างไรว่าควรจะกำหนดเรื่องใดไว้ก่อน ก็ขอเรียนให้ทราบว่าผมซ้อมเขียนโครงงานทำนองนี้มาหลายโครงงานแล้ว ดังที่นำมาเสนอในไมโครฯ เป็นลำดับมา จึงพอมองออกว่าควรกำหนดเรื่องใดไว้ก่อนบ้าง (แต่ก็ต้องแก้อยู่หลายครั้งเหมือนกัน) กล่าวคือ

ข้อกำหนดของ Console (รูป A1)

เป็นส่วนข้อกำหนดที่จะใช้ในยูนิต CONSOLE (Console Command) แต่ก็มีที่ใช้ในยูนิตอื่นด้วยเหมือนกัน จึงนำมากำหนดไว้เป็นส่วนกลาง (global) ประกอบด้วย ค่าคงตัว แบบข้อมูล และตัวแปรดังนี้

- StreamT เป็นชื่อแบบข้อมูล ที่กำหนดเรียกเป็น สตรีม เพื่อให้แตกต่างกับสตริง ใช้สำหรับกำหนดตัวแปรพอยน์เตอร์ไปยังบัฟเฟอร์เพื่อเก็บค่าที่

อ่านจากคีย์บอร์ด และสำหรับเก็บโปรแกรมย่อยในภาษาล็อกศูนย์ ที่ให้ยาว 250 อักขระก็เป็นแต่เพียงการกำหนดเท่านั้น ความยาวจริงจะกำหนดในคำสั่ง GetMem

หมายเหตุ

1. คำว่า "ผู้ใช้" ที่ผมจะกล่าวถึงต่อไปในบทความนี้ หมายถึงผู้ที่จะเขียนโปรแกรมภาษาล็อกศูนย์ของเรา (เช่นเดียวกันกับที่เราเป็นผู้ใช้ของเทอร์โบปาสคาล)

2. คำว่า "โปรแกรมย่อยในภาษาล็อกศูนย์" ที่ผู้ใช้หรือคุณและผมเขียนขึ้น ต่อไปเพื่อความกะทัดรัดจะใช้คำว่า "โปรแกรมเต่า" แทน

- proc เป็นตัวแปรแบบอะเรย์ขนาด 2001 ช่อง ในแบบอะเรย์ของเรคอร์ดที่มีความยาว (len) และพอยน์เตอร์ (pntr) ไปยังสตรีม โดยที่ผมจะให้ช่องศูนย์ (พอยน์เตอร์ในช่องศูนย์ของอะเรย์ proc) ชี้ไปที่บัฟเฟอร์ซึ่งจะเก็บคำสั่งที่ผู้ใช้ป้อน และเป็นบัฟเฟอร์ของเอดิเตอร์ด้วย ส่วนช่องอื่น ๆ จะชี้ไปยังสตรีมที่เก็บโปรแกรมเต่าช่องละ 1 โปรแกรม นั่นคือผู้ใช้จะสร้างโปรแกรมเต่าได้ไม่เกิน 2,000 โปรแกรม หากต้องการเปลี่ยนแปลงให้แก้ที่ค่าคงตัว MAXPRO

- wrd เป็นตัวแปรสำหรับใช้งานทั่วไปแบบสตริง 255 อักขระ เพื่อเก็บคำในภาษาล็อกศูนย์ 1 คำ ที่ให้ยาวมากเช่นนี้เพื่อไม่ต้องจำกัดความยาวของคำที่ผู้ใช้กำหนดขึ้น และเพื่อใช้งานอื่นด้วย

- IdenT ผมไม่ทราบว่าในภาษาโลโกได้จำกัดขนาดของคำมาตรฐานหรือไม่ แต่ในล็อกศูนย์ผมคิดว่าควรจำกัดขนาดไว้บ้างเพื่อความรวดเร็วในการค้นหาคำ จึงกำหนดให้คำมาตรฐาน (ไม่เกี่ยวกับผู้ใช้) และชื่อตัวแปรให้ยาวไม่เกิน 19 อักขระ

- ตัวแปร err, perr และ cerr เพื่อบ่งสถานะการทำงานของโปรแกรม โดยให้ err บ่งสถานะว่ามีข้อผิดพลาดเกิดขึ้นเมื่อสถานะไม่ใช่ศูนย์ ที่ให้เป็นเลขจำนวนเต็มเพื่อให้บอกได้ว่าผิดในเรื่องอะไร ส่วน perr และ cerr เพื่อให้ทราบที่ผิดในโปรแกรมเต่า โดยที่ perr จะบอกว่าเป็นโปรแกรมย่อยที่เท่าไรจากต้นอะเรย์ proc ส่วน cerr บ่งว่าเป็นอักขระตัวที่เท่าไรของโปรแกรมเต่านั้น

- ตัวแปร fch และ lch มีลักษณะเป็นการบ่งสถานะเช่นกัน โดย fch เก็บอักขระตัวแรกของคำ ส่วน lch เก็บอักขระตัวสุดท้ายของคำ

- ตัวแปร fin ไว้จบการทำงานของล็อกศูนย์ ส่วน datum ไว้หมายตำบลที่ในฮีพดังจะกล่าวต่อไป

- ค่าคงตัว ที่กำหนดต่อไปเป็นพวกรหัสควบคุมต่าง ๆ

- Cons เป็นอะเรย์ของคำที่ให้ผู้ใช้ใช้ในการสั่งเครื่องดังที่กล่าวมาแล้ว ที่เพิ่มขึ้นมาคือ cls ให้ลบจอภาพกับ park ให้เต่าอยู่ ณ ตำแหน่งเมื่อเริ่มต้น เป็นคำที่สร้างขึ้นใช้ชั่วคราวต่อไปต้องยกเลิก ส่วน new หมายถึงให้เริ่มต้นใหม่หมดโดยไม่ต้องออกจากล็อกศูนย์ คำพวกนี้เมื่อทำให้เป็นภาษาไทยให้แก้ ณ ตรงนี้แห่งเดียว

จอภาพ (รูป A2)

เป็นข้อกำหนดชั่วคราวเมื่อยังไม่ติดตั้งกราฟิกเท่านั้น ต่อไปต้องยกเลิกทั้งหมดและได้เคยกล่าวถึงมาหลายครั้งแล้วจึงของดกล่าวในรายละเอียด มีส่วนที่จะกล่าวถึงคือผมได้กำหนดจอภาพไว้เป็น 3 หน้าต่างคือ

- main เต็มจอภาพเพื่อใช้ในการอีดิต
- turt คือส่วนที่ให้เต่าเดิน
- user คือส่วนที่ให้ผู้ใช้ป้อนคำสั่ง

ดังที่ได้นำมากำหนดค่าคงตัวให้แก่ wnd และได้กำหนดตัวแปร temscrn ไว้สำหรับเก็บหน้าต่าง turt กับ user เมื่อเปิดใช้ main

ข้อกำหนดของ Turtle (รูปที่ A3)

เป็นส่วนข้อกำหนดที่จะใช้ในยูนิต

A1

```
UNIT  Log00;

Interface
uses   crt;


(*-------------------   Console  declaration   -------------*)

CONST    MAXSTRM = 250;  MAXPRO = 2000;  MAXBUF = 2048;
TYPE     StreamT = ARRAY[1..MAXSTRM] OF Char;
         PntrT   = ^StreamT;
         ProcT   = ARRAY[0..MAXPRO] OF
                     record  len : Word;  pntr : PntrT  end;
VAR      proc  : ProcT;  pm : Word;
         wrd   : String;

TYPE    Str79 = String[79];  Str12 = String[12];
         PathT = Str79;  FiNameT = Str12;
         IdenT = String[19];
VAR      err,perr,cerr : Integer;  fch,lch : Char;  fin : Boolean;
         datum : Pointer;

CONST    CR = #13;  FF = #12;  LF = #10;  ESC = #27;   BS = #8;
         ENDFI = #26;

TYPE     ConT = (load,list,save,cls,prk,renew);
CONST    Cons  : ARRAY[ConT] OF IdenT =
           ('load','list','save','cls','park','new');
```

ที่เกี่ยวข้องกับ TurtleCommand ซึ่งส่วนมากจะใช้งานในอินเตอร์เพรตเตอร์ แต่มีส่วนที่ใช้กับยูนิตอื่นด้วย จึงนำมาไว้ในยูนิต LOG00 กล่าวคือ

- ค่าคงตัว ได้กำหนดค่าคงตัวของเครื่องหมายต่าง ๆ และเซตของอักขระเพื่อความสะดวกในการเขียนโปรแกรม ค่าคงตัวต่อไปหมายถึงตัวเต่าแต่เป็นรูปดอกจัน (ผมจึงเรียกเป็นแมงมุม) กับรอยเท้าที่เต่าเดินไป ดังแสดงในรูปที่ 3 ส่วน nofoot ไว้ลบรอยเท้า และเมื่อให้เต่าเดินโดยไม่มีรอยเท้าสำหรับ xadj กับ yadj สำหรับปรับแต่งจุดโคออร์ดิเนตเพื่อให้ผู้ใช้ถือว่าจุด (0,0) อยู่ที่กึ่งกลางหน้าต่าง turt (ค่าคงตัวกลุ่มหลังนี้จะต้องเปลี่ยนใหม่เมื่อติดตั้งกราฟิก)

- ตัวแปร head เพื่อเก็บทิศทางของเต่ามี rx และ ry เป็นตำบลที่ ส่วน col และ row เป็นตำบลที่เช่นกัน แต่เมื่อจุดโคออร์ดิเนต (1,1) อยู่ที่มุมบนซ้ายของจอภาพ fast เมื่อให้เดินโดยไม่ทิ้งรอยเท้าเป็นการเดินเร็ว turtle เพื่อให้แสดงภาพเต่าหรือไม่ให้แสดง step เพื่อการเดินโดยทิ้งรอยเท้าหรือไม่ทิ้ง ส่วน mypi เพื่อเก็บค่า pi/180 ไว้เพื่อไม่ต้องคำนวณกันทุกครั้งที่ใช้คำนี้

- DataT ในภาษาล็อกศูนย์มีแบบข้อมูลอยู่ 4 แบบคือ เลขจำนวน อักขระ คำ (คือแถวของอักขระที่ไม่เว้นวรรค) และลิสต์ ซึ่งอาจประกอบด้วย เลขจำนวน อักขระ หรือคำ ซึ่งจะผสมกันอย่างไรก็ได้จำนวนหนึ่งอยู่ภายในเครื่องหมายวงเล็บก้ามปู

ในการเก็บข้อมูลผมจะเก็บอักขระและเลขจำนวนซึ่งผู้ใช้ป้อนเข้ามาเป็นแถวของตัวเลขไว้เป็นค่าของมัน คือเป็นอักขระและเลขจำนวน ส่วนข้อมูลอื่น ๆ จะเก็บเป็นพอยน์เตอร์ไปยังข้อมูลนั้น ๆ ในโปรแกรมเต่า ดังที่กำหนดเป็นแบบข้อมูลให้แก่ DataT

ทั้งนี้เพราะเลขจำนวนใช้บ่อยมาก จะได้ไม่ต้องแปลงค่าซ้ำแล้วซ้ำอีก ทำให้เสียเวลา (หมายถึงเมื่อใช้ตัวแปร) ส่วนข้อมูลแบบอื่นมีที่ใช้น้อย แต่จุดประสงค์หลักคือเพื่อไม่ต้องกำหนดขนาดของคำและจำนวนสมาชิกของลิสต์ (ไม่ทราบเหมือนกันครับว่าผมออกแบบผิดหรือไม่)

สำหรับแบบข้อมูลที่กำหนดให้แก่ DataT ได้กำหนดเป็นเรคอร์ดที่มีฟิลด์ varient เพียงฟิลด์เดียว โดยไม่มี tag ฟิลด์ เพราะผมต้องการประหยัดเวลาเป็นอย่างมากและคิดว่าไม่น่าจะมีปัญหา (หากมีปัญหาก็ต้องย้อนกลับมากำหนด tag ฟิลด์กันใหม่)

- VarT เพื่อการกำหนดหน่วยความจำสำหรับเก็บตัวแปรในภาษาล็อกศูนย์ ประกอบด้วยชื่อตัวแปรกับข้อมูล

- vars และ gvar เป็นหน่วยความจำที่จะเก็บตัวแปรที่เป็นโลคอลและโกลบอลตามลำดับ สำหรับโลคอลใช้เพื่อการผ่านค่าผมน่าจะตั้งชื่อว่า stack มากกว่า สำหรับโกลบอลเพื่อเก็บตัวแปรที่

A2

```
(*--------------------  SCREEN declaration ------------------*)

CONST   norm = $07;  rev = $70;  inten = $0F;  blink = $87;
        undlne = $01;
TYPE    ScreenT = ARRAY[1..25,1..80] OF
                          RECORD  ch : Char; attr : Byte  END;
        TranScrT = ARRAY[1..25,1..80] OF Word;
        WndNme   = (main,turt,user);
        WindowT  = ARRAY[1..4] OF Byte;
        CursorT  = ARRAY['x'..'y'] OF Byte;
        WndS     = ARRAY[WndNme] OF WindowT;
        CurS     = ARRAY[WndNme] OF CurSorT;
CONST   wnd : WndS = ((1,1,80,25), (1,1,80,21), (1,23,80,25));
        cdum : CursorT = (1,1);
VAR     scrn : ScreenT ABSOLUTE $B000:$0000;
        transcrn : TranScrT ABSOLUTE $B000:0000;
        temscrn  : TranScrT;  { ABSOLUTE $B000:1000;}
        cur      : CurS;  lwnd : WndNme;
```

A3

```
(*-------------------- TURTLE declaration -------------------*)

CONST  SPACE = ' ';   OPNBLK = '[';   CLSBLK = ']';
       QUOT  = '"';   LFTPAR = '(';   RGTPAR = ')';
       VARSIGN = ':';

       Skip  : SET OF Char = [SPACE,CR,LF];
       Skippls  : SET OF Char = [SPACE,CR,LF,CLSBLK,RGTPAR,#0];
       OPNSGN   : SET OF Char = [QUOT,OPNBLK,VARSIGN];

       tur = '*';  foot = #9;  nofoot = SPACE;
       Xadj = 40;  Yadj = 11;
VAR     head, rx, ry : Real;  col, row : Integer;
        fast,turtle : Boolean;  step : Char;  mypi : Real;

CONST   MAXVAR = 1000;  MAXGVAR = 100;
TYPE    DataT  = RECORD
                 CASE  Integer  OF
                  0 : (num  : Real);
                  1 : (ch   : Char);
                  2 : (pntr : PntrT);
                   END;
        VarT = RECORD  name : IdenT;  data : DataT  END;
VAR     vars   : ARRAY[1..MAXVAR] OF Vart;
        gvar   : ARRAY[1..MAXGVAR] OF VarT;  gm : Word;

TYPE    ComT = (penup,pendn,draw,nodraw,home,nl,clean,nop,
                print,fwrd,bwrd,lft,rght,none);
        ComName = ARRAY[ComT] OF IdenT;
CONST   com : ComName =
                ('pu','pd','draw','nodraw','home','nl','cs','nop',
                 'pr', 'fd','bk','lt','rt','');
```

กำหนดด้วยคำสั่ง Make โดยที่ในภาษาโลโกจะคงค่าไว้ตลอดเวลาที่รันโลโก ซึ่งผมไม่เห็นด้วยเลย เห็นว่าควรให้คงอยู่ตลอดเวลารันโปรแกรมของผู้ใช้หนึ่ง ๆ เท่านั้น แต่ไม่กล้าเปลี่ยนแปลง เพราะดังที่กล่าวแล้วว่า ภาษาโลโกออกแบบโดยนักคอมพิวเตอร์ศาสตร์และนักศึกษาศาสตร์

- ตัวแปร gm เพื่อชี้ไปยังตัวแปรหลังสุดใน gvar

- com คือคำในภาษาล็อกศูนย์ที่เป็นโพรซีเดอร์ (ในความหมายของภาษาปาสคาล) คำพวกนี้ผมกำหนดไว้เพียงเพื่อให้เขียนอินเตอร์เพรตเตอร์ได้เท่านั้น จะต้องเปลี่ยนแปลงใหม่ ทั้งเมื่อเขียน Library และเมื่อติดตั้งภาษาไทย โดยมีเงื่อนไขว่าต้องให้ nop อยู่หลังสุดของโปรแกรมย่อยที่ไม่ต้องการอาร์กิวเมนต์ด้วยกัน

หมายเหตุ คำภาษาอังกฤษที่เป็นแบบข้อมูลของ ComT เช่น penup เมื่อเปลี่ยนเป็นภาษาไทยไม่จำเป็นต้องเปลี่ยนคำนี้ แต่ต้องเปลี่ยนคำที่จะใช้ในโปรแกรมเต่า คือที่ให้ค่าประจำแก่ตัวแปร com เช่น 'pu' อาจเปลี่ยนเป็น 'ยกปากกา' หรือ 'ยก'

คำที่จะใช้ในโปรแกรม(รูป A4)

นอกจากคำที่เป็นโพรซีเดอร์ตามที่กำหนดให้แก่ com ยังมีคำอีกจำนวนหนึ่งซึ่งต้องกำหนดขึ้นเพื่อใช้สั่งเครื่องและใช้ในโปรแกรมของผู้ใช้ กล่าวคือ

*** คำสงวน (reserved word)** เพื่อสั่งเครื่องและควบคุมการทำงานของโปรแกรม ได้แก่

- TOO แทนคำว่า to เมื่อผู้ใช้ต้องการป้อนโปรแกรม

- QUIT แทนคำว่า quit เมื่อผู้ใช้ต้องการยุติการทำงานในล็อกศูนย์

- REPEATT แทนคำว่า repeat เทียบเท่ากับ FOR ในภาษาปาสคาล

A4

```
(*--------------- English Implement ----------------------*)
CONST  { Console Command }

       { reserved words }
  TOO      : IdenT = 'to';
  QUIT     : IdenT = 'quit';
  REPEATT  : IdenT = 'repeat';
  IFF      : IdenT = 'if';
  MAKE     : IdenT = 'make';
  ENDP     : IdenT = 'end;';
  STOP     : IdenT = 'stop';

       { standard identifier }
  heading  : IdenT = 'heading';
  Xcor     : IdenT = 'Xcor';
  Ycor     : IdenT = 'Ycor';
  ReadCh   : IdenT = 'rc';

       { others }
  ENDS = ';';
  NUMCHAR = 6;
  PlsEnd   : String[NUMCHAR] = ' end; ';
```

- IFF แทนคำว่า if เทียบเท่ากับ IF ในภาษาปาสคาล

- MAKE แทนคำว่า make เพื่อการกำหนดและให้ค่าตัวแปร

- ENDP แทนคำว่า end; เพื่อแสดงว่าจบโปรแกรมเต่าหนึ่ง ๆ

- STOP แทนคำว่า stop เมื่อผู้ใช้ต้องการให้ยุติการเรียกตนเอง

*** ฟังก์ชัน (Function)** เทียบเท่ากับฟังก์ชันในภาษาปาสคาล และต้องกำหนดไว้ตรงนี้แทนที่จะกำหนดในไลบรารี ผมกำหนดไว้เพียงเพื่อการเขียนอินเตอร์เพรตเตอร์เท่านั้น และตัวฟังก์ชันจริงจะอยู่ในยูนิต Cal

- heading แทนคำว่า heading เพื่อให้ค่าทิศทางของเต่า

- Xcor และ Ycor แทนคำว่า Xcor และ Ycor เพื่อให้ตำบลที่ของเต่าในแกน X และ Y ตามลำดับ

- ReadCh แทนคำว่า rc ให้ค่าคีย์เป็นอักขระ 1 ตัว เทียบเท่ากับ Read Key ในเทอร์โบปาสคาล

คำพวกนี้จะต้องเปลี่ยนเป็นภาษาไทยต่อไป เช่น คำว่า ENDP อาจให้แทนคำว่า "จบ;" ผมจึงนำคำที่จะต้องเปลี่ยนเมื่อติดตั้งภาษาไทยมารวมไว้ด้วยกันในหัวข้อ others คือ

- ENDS เป็นอักขระ ;

- PlsEnd เป็นสตริงที่จะนำไปต่อท้ายสตริมที่ผู้ใช้ป้อนดังจะกล่าวต่อไป

โปรแกรมสนับสนุน

ถัดจากการกำหนดแบบข้อมูล ค่าคงตัว และตัวแปรแล้วเป็นการกำหนดส่วนหัวของโปรแกรมย่อย (ที่ผมถือว่าเป็นโปรแกรมสนับสนุนหรือโปรแกรมย่อยใช้งานทั่วไป) ตามที่เทอร์โบปาสคาลให้กำหนดดังรูป A5 ส่วนตัวโปรแกรมจริงอยู่ในรูป A6 ถึง A10 ได้แก่

Init (รูป A6)

เพื่อเตรียมค่าต่าง ๆ ที่ผมแยกมาเขียนรวมกันไว้ เพื่อให้โพรซีเดอร์ RenewPrg เรียกใช้ได้ด้วย (โพรซีเดอร์นี้จะทำงานเมื่อผู้ใช้ป้อนคำว่า new) ขอให้สังเกตว่า Init สามารถเรียกใช้โปรแกรมย่อยที่อยู่หลังมันได้ ทั้งนี้เพราะเราได้กำหนดส่วนหัวของโปรแกรมย่อยเหล่านี้ไว้ในส่วน Interface จึงกลายเป็นชื่ออ้างถึงล่วงหน้า (forward reference) ไปในตัว

OpenWindow (รูป A7)

เพื่อการเปิดใช้หน้าต่างที่ต้องการ

(main, turt และ user) โปรแกรมนี้ผมเขียนอย่างง่าย ๆ เพราะถึงอย่างไรก็ต้องเปลี่ยนแปลงใหม่อยู่แล้ว

ReadStream กับ ReadOneStream (รูป A8)

โปรแกรมที่จะอ่านคำสั่งที่ผู้ใช้ป้อนแท้ ๆ คือ ReadOneStream แต่เพื่อไม่ต้องเขียนตัวแปรที่ใช้ในการผ่านค่ายาว ๆ ดังที่เห็นใน ReadStream จึงให้เรียก Readstream ก่อนด้วยค่า px (หมายถึงช่องที่ px ในอะเรย์ proc) ซึ่ง ReadStream จะผ่านค่าพอยน์เตอร์คือแอดเดรสถึงหน่วยความจำที่กำหนดเรียกเป็นสตรีมไปให้ ส่วนใน ReadOneStream รับการผ่านค่าในแบบอ้างถึง จึงสามารถอ้างถึงหน่วยความจำนี้ได้ในแบบอะเรย์ของอักขระธรรมดา

การทำงานของ ReadOneStream จะอ่านคีย์เข้ามาครั้งละ 1 อักขระ นำไปเก็บในสตรีม แต่ก่อนเก็บจะตรวจดูก่อนว่าถ้าเป็น CR (คือเมื่อผู้ใช้กดคีย์ Enter) ให้ปิดท้ายสตรีมด้วยรหัสอักขระ #0 ถ้าเป็นคีย์ back space ให้ลบภาพอักขระนั้นบนจอภาพ และยกเลิกค่านั้นในสตรีม ทำงานเป็นวงรอบอยู่เช่นนี้จนกว่าผู้ใช้จะกดคีย์ Enter ถือว่าจบการป้อน

GetToken กับ GetOneToken (รูป A9)

ในการดำเนินการกับคำสั่งและโปรแกรมเต่าซึ่งจะขอรวมเรียกว่าข้อมูล จะดำเนินการโดยอ่านข้อมูลเข้ามาทีละคำ ซึ่ง GetOneToKen จะดำเนินการในเรื่องนี้โดย จะอ่านข้ามวรรคและ CR กับ LF เมื่อถึงอักขระตัวอื่นจะนำมาตรวจว่าเป็นพวกเครื่องหมายนำคือเครื่องหมายคำพูด เครื่องหมายวงเล็บปีกกาเปิด เครื่องหมาย : ซึ่งแสดงว่าเป็นตัวแปรหรือไม่ ถ้าใช่ให้เก็บอักขระนี้ในตัวแปร fch เพื่อเก็บไว้ใช้เป็นเงื่อนไขและอ่านข้ามอักขระนี้ไป

จากนี้จะเข้าทำงานในวงรอบ WHILE อ่านอักขระทีละตัว ให้แก่ตัวแปรสตริง wrdจนกว่าจะพบอักขระปิดท้ายคือ วรรค รหัส CR และ LF วงเล็บก้ามปูปิด วงเล็บเปิด หรือรหัส #0 ตัวใดตัวหนึ่ง ถือว่าจบคำ ให้ค่าความยาวของสตริง เก็บรหัสตัวสุดท้ายนี้ไว้ใน lch เพื่อไว้ใช้เป็นเงื่อนไข แล้วเพิ่มค่า cx ซึ่งชี้อักขระในสตรีมขึ้นหนึ่งถ้าอักขระตัวสุดท้ายไม่ใช่ #0 นอกจากนี้ให้ตรวจอีกว่าถ้าอักขระตัวสุดท้ายเป็น CR หรือ LF จะให้ lch เป็นวรรค เพราะถือ

A5

```
PROCEDURE  Init;
PROCEDURE  OpenWindow(nme : WndNme);
PROCEDURE  ReadOneStream(VAR astr : StreamT);
PROCEDURE  ReadStream(px : Word);
PROCEDURE  GetOneToken(var astr : StreamT; var cx : Word;
                                    var wrd : String);
PROCEDURE  GetToken(var px,cx : Word; var wrd : String);
PROCEDURE  MarkErr(px,cx : Word; x : Integer);
PROCEDURE  WritePntr(ptr : Pointer);
```

A6

```
Implementation

PROCEDURE  Init;
VAR  i : Byte;
BEGIN
  OpenWindow(main);  ClrScr;
  cur[main] := cdum;  cur[turtl := cdum;  cur[user] := cdum;
  FOR  i := 1 TO 80  DO  scrn[22,i].ch := #205;
  head := 0.0; rx := 0.0;  ry := 0.0;  col := Xadj;  row := Yadj;
  step := foot;  fast := FALSE;
  pm := 1; gm := 1;  mypi := Pi / 180;  fin := FALSE;
  scrn[row,col].ch := tur;  lwnd := main;  temscrn := transcrn;
END;
```

A7

```
PROCEDURE  OpenWindow(nme : WndNme);
BEGIN
  cur[lwnd,'x'] := WhereX;  cur[lwnd,'y'] := WhereY;
  Window(wnd[nme,1], wnd[nme,2], wnd[nme,3], wnd[nme,4]);
  GotoXY(cur[nme,'x'], cur[nme,'y']);  lwnd := nme;
END;
```

A8

```
PROCEDURE  ReadOneStream(VAR astr : StreamT);
VAR     i : Integer;   ch : Char;  x,y : Byte;
BEGIN
  while  KeyPressed  do  ch := ReadKey;
  i := 1;
  REPEAT
    ch := ReadKey;
    IF  ch = CR  THEN  astr[i] := #0
    ELSE IF  i = MAXSTRM  THEN  Write(^G)
    ELSE IF  ch >= SPACE  THEN
      BEGIN  Write(ch);  astr[i] := ch;   Inc(i)  END
    ELSE IF  (ch = BS) and (i > 1)  THEN
      BEGIN  Write(ch);  y := WhereY;  x := WhereX;
             Write(' ');  GotoXY(x,y);  Dec(i)    END;
  UNTIL  ch = CR;
  while  KeyPressed  do  ch := ReadKey;
END;


PROCEDURE  ReadStream(px : Word);
BEGIN  ReadOneStream(proc[px].pntr^);  END;
```

A9

```
PROCEDURE  GetOneToken(var astr : StreamT; var cx : Word;
                                           var wrd : String);

VAR  id : Word;
BEGIN
  while  astr[cx] in skip  do  inc(cx);
  id := 1;  fch := astr[cx];
  if  fch in OPNSGN  then  inc(cx);
  while  not (astr[cx] in skippls)  do
    begin  wrd[id] := astr[cx];  inc(id);  inc(cx)  end;
  wrd[0] := Chr(id-1);  lch := astr[cx];
  if  lch <> #0  then  inc(cx);
  if  lch in skip  then  lch := SPACE;
END;


PROCEDURE  GetToken(var px,cx : Word; var wrd : String);
BEGIN  GetOneToken(proc[px].pntr^,cx,wrd);  END;
```

A10

```
PROCEDURE  MarkErr(px,cx : Word; x : Integer);
BEGIN
  perr := px;  cerr := cx;  err := x;
END;
{$F+}  FUNCTION  HeapFunc(size: Word) : Integer;  {$F-}
       BEGIN  HeapFunc := 1  END;


BEGIN
  GetMem(proc[0].pntr, MAXBUF);  Mark(datum);
  HeapError := @HeapFunc;
  Init;
END.
```

ว่าการขึ้นบรรทัดใหม่เทียบเท่ากับการเว้นวรรค

เมื่อจบโปรแกรมจะได้ข้อมูลที่เป็นคำในตัวแปร wrd และค่าอินเด็กซ์ของสตรีมคือ cx ที่เพิ่มขึ้นส่งกลับไป ทำให้โปรแกรมที่เรียกมาสามารถอ่านข้อมูลเป็นคำนำไปตีความต่อเนื่องกันไปได้จนสุดสตรีม สตรีมหนึ่ง ๆ อาจเป็นคำสั่งของผู้ใช้ 1 คำ หรือทั้งบรรทัด หรือเป็นโปรแกรมย่อยของผู้ใช้ 1 โปรแกรม ซึ่งอาจเรียกใช้โปรแกรมย่อยอื่นในสตรีมอื่นต่อไปอีกก็ได้

ซึ่งเมื่อผมได้วางแนวความคิดเช่นนี้แล้วก็เขียนโปรแกรมเดินหน้าเรื่อยไป แต่เมื่อนำโปรแกรมมาทบทวนเห็นว่ามีการทำงานซ้ำซ้อนในการเทียบคำ วิธีที่ดีกว่าคือใช้ตัวแปรพอยน์เตอร์ชี้ไปที่ต้นคำ ซึ่งจะทำให้ไม่ต้องกำหนดความยาวของตัวแปรด้วย แต่ข้อเสียคือต้องแก้โปรแกรมหลายแห่งจึงขอพักไว้ก่อนครับ

MarkErr (รูป A10)

โพรซีเดอร์นี้จะเก็บ px ซึ่งหมายถึงอักขระที่ในโปรแกรมเด่านี้ กับค่า err ซึ่งเป็นรหัสบ่งถึงข้อผิดพลาดไว้ในตัวแปรดังกล่าวมา

ซึ่งผมมีแนวความคิดว่าหากมีข้อผิดพลาดเกิดขึ้น จะให้เข้าเอดิเตอร์โดยอัตโนมัตินำโปรแกรมที่ผิดออกแสดงแล้วให้เคอร์เซอร์อยู่ตรงคำที่ผิด และให้แสดงข้อความบ่งถึงข้อผิดพลาดที่บรรทัดบนสุด แต่ยังไม่ได้เขียนเอดิเตอร์จึงยังไม่มีการดำเนินการใดต่อ

HeapFunc (รูป A10)

เป็นฟังก์ชันที่ดำเนินการแทนโปรแกรมจัดการในเรื่องฮีพ เช่น เมื่อเรากำหนดหน่วยความจำในฮีพหากหน่วยความจำไม่พอ จะเกิด error และยุติการรัน จึงสร้างฟังก์ชันนี้เพื่อดำเนินการแทน โดยให้ค่าเป็น 1 เพื่อไม่ต้องยุติการรัน

โปรแกรมหลักของ LOG00 (รูป A10)

เมื่อใช้งานยูนิต เทอร์โบปาสคาลจะรันโปรแกรมที่อยู่ในส่วนถ้อยแถลงของยูนิตก่อน สำหรับยูนิต LOG00 ได้ให้จองเนื้อที่ขนาด 2048 ไบต์ไว้ โดยให้พอยน์เตอร์ใน proc[0] ชี้มาที่หน่วยความจำนี้ ซึ่งผมจะใช้เก็บคำสั่งที่ผู้ใช้ป้อนเป็นสตรีม และเมื่อเรียกใช้เอดิเตอร์จะให้เป็นบัฟเฟอร์ ซึ่งผมคิดว่าโปรแกรมเด่าหนึ่ง ๆ ไม่น่าจะยาวเกิน 2048 ไบต์

ถัดจากนั้นได้ให้ Mark แอดเดรสเหนือหน่วยความจำนี้ ทั้งนี้เพื่อที่เมื่อผู้ใช้ใช้คำสั่ง new ผมจะให้ยกเลิก (Dispose) หน่วยความจำที่จองในภายหลังทั้งหมด แต่คงหน่วยความจำหน่วยนี้ไว้

ถ้อยแถลงถัดไปเป็นการให้ค่าตัวแปร HeapError ด้วยค่าแอดเดรสของฟังก์ชัน HeapFun เพื่อให้ทำหน้าที่แทนโปรแกรมจัดการในเรื่องฮีพดังกล่าวแล้ว ท้ายสุดคือให้เตรียมค่าต่าง ๆ ด้วย Init

ขอให้ตั้งชื่อไฟล์ว่า LOG00.PAS ป้อนโปรแกรมนี้แล้วคอมพายล์จนผ่านเก็บซอร์สโปรแกรมลงดิสค์ไว้

โปรแกรมหลัก

ผมคิดว่าก่อนที่จะกล่าวถึงยูนิตอื่นต่อไปหากนำเรื่องราวของโปรแกรมหลักมากล่าวถึงก่อนจะสะดวกต่อการทำความเข้าใจ ดังแสดงในรูป M0 ซึ่งทำเป็นส่วนของโปรแกรมแยกเก็บไว้ในอีกไฟล์หนึ่ง

ตอนแรกเป็นการกำหนดตัวแปรเพื่อใช้ในโปรแกรมหลักโดยเฉพาะ ส่วนที่เขียนไว้เป็น Other Unit คือโปรแกรมย่อยในยูนิตอื่นที่ยังไม่ได้สร้าง แต่จะมีการเรียกใช้ในโปรแกรมหลัก จึงให้เป็น begin end; ไว้เท่านั้น

Mϕ

```
{$M  65520,65536,655360}
{$V-}
PROGRAM  Log0;
uses  crt,Log00;

VAR     wrd : String;  px,cx : Word;  def  : ConT;  ch : Char;

(*-----------------     Other  Unit     -----------------*)

FUNCTION   ConsComm(var wrd : String; var def : ConT) : Boolean;
begin  end;
PROCEDURE  DoConsole(def : Cont; px,cx : Word);
begin  end;
procedure  DoTurtleComm(var wrd : String; px,cx,v1,v2 : Word);
begin  end;
procedure  DoEdit(px : Word);
  begin  end;

(*.....................................................*)

BEGIN  { MAIN }
  OpenWindow(user);

  REPEAT
    err := 0;  perr := 0;  cerr := 0;  px := 0;
    Write('<');
    while  KeyPressed  do  ch := ReadKey;
    ReadStream(px);  cx := 1;
    GetToken(px,cx,wrd);
    IF  wrd = ''  THEN  WriteLn('>')
    ELSE IF  wrd = TOO  THEN  DoEdit(px)
    ELSE IF  wrd = QUIT  THEN  fin := TRUE
    ELSE IF  ConsComm(wrd,def)  THEN  DoConsole(def,px,cx)
    ELSE  DoTurtleComm(wrd,px,cx,1,1);

    IF  wrd <> ''  THEN  Write('> ');
    IF  (err <> 0) AND (err <> 100)  THEN
       WriteLn('ERROR = ',err);
  UNTIL  fin;

writeln; write(MemAvail,' end of program press ENTER ');  readln;
Write('clear screen (y/n) ');  ch := Upcase(ReadKey);
if  ch = 'Y' then  begin  Window(1,1,80,25);  ClrScr end;
END.
```

หากดูประกอบกับ uses จะเห็นว่าโปรแกรมหลักตามรูป M0 นี้ จุดประสงค์เพื่อทดสอบยูนิต LOG00 เท่านั้น และจะมีการเปลี่ยนแปลงเมื่อติดตั้งยูนิตอื่น ๆ

โปรแกรมหลักเริ่มด้วยเปิดใช้หน้าต่าง user แล้วเข้าทำงานในวงรอบ repeat ซึ่งการทำงานในวงรอบนี้แต่ละครั้ง จะอ่านคำสั่งที่ผู้ใช้ป้อนเป็นสตรีมมาตีความ โดยแยกคำแรกด้วย GetToken มาตรวจว่า

- ถ้าเป็นสตรีมว่าง คือผู้ใช้กดคีย์ Enter เพียงคีย์เดียว ให้ขึ้นบรรทัดใหม่
- ถ้าเป็นคำว่า 'to' ให้เรียกใช้ DoEdit ซึ่งจะไปเรียกใช้ Editor ต่ออีกทอดหนึ่ง
- ถ้าเป็นคำว่า 'quit' จะให้ fin เป็นจริงเพื่อออกจากวงรอบ repeat และยุติการรัน
- ถ้าเป็นคำนอกจากนี้ ให้ตรวจว่าเป็นคำสั่งเครื่อง เช่น load หรือไม่ ถ้าใช่ให้ดำเนินการด้วย Doconsole
- ถ้าไม่ใช่อีกก็จะต้องเป็นคำสั่งเต่า ให้ดำเนินการด้วย DoTurtleComm ซึ่งถ้าหากไม่ใช่อีก DoTurtleComm จะให้ค่า err ถ้าหากใช่ก็จะดำเนินการต่อไป

เมื่อมาถึงท่อนหลังแสดงว่าได้ดำเนินการตามคำสั่งของผู้ใช้เสร็จแล้ว ผมได้ให้ปิดท้ายด้วยเครื่องหมาย > เพื่อให้เป็นที่สังเกต ทั้งนี้เพื่อไม่ต้องให้ขึ้นบรรทัดใหม่ เพราะผมมีความเห็นว่าสำหรับภาษาไทยเรา เนื่องจากต้องออกจอภาพหลายระดับอยู่แล้ว จึงควรจะใช้เนื้อที่ในแต่ละบรรทัดบนจอภาพให้มากที่สุด

จากนี้จะตรวจว่าคำสั่งซึ่งจะรวมไปถึงโปรแกรมเต่าด้วยว่ามีที่ผิดหรือไม่ ถ้ามีให้แสดงที่ผิดด้วย DispError แต่เนื่องจากเรายังไม่ได้เขียน Editor จึงเพียงแต่แสดงว่าผิดด้วยเรื่องอะไรเท่านั้น

ตอนท้ายของโปรแกรมหลักเป็นส่วนที่ผมใช้ในการพัฒนาโปรแกรมนี้เท่านั้น ไม่เกี่ยวกับภาษาล็อกศูนย์

ขอให้คุณตั้งชื่อไฟล์ว่า MAIN0.PAS ป้อนโปรแกรมนี้แล้วคอมพายล์ Compile/Build จนผ่าน ซึ่งจะเป็นการสร้างไฟล์ LOG00.TPU ด้วย และที่ผมกล่าวในตอนแรกว่าสร้าง MAIN0 เพื่อทดสอบยูนิต LOG00 นั้น การดำเนินการถึงขั้นตอนนี้เป็นเพียงตรวจหาที่ผิดใน MAIN0 เท่านั้น

ซอฟต์เมล์

ผมได้มอบโปรแกรมสำหรับโครงงานนี้ไปจนถึง MAIN2 มาสำหรับซอฟต์เมล์ด้วยแล้ว ซึ่งหากคุณคอมพายล์ (ขอให้คอมพายล์ในแบบ Compile/Build) จะสามารถสั่งเต่าให้เคลื่อนไหวได้ เพราะผมได้เขียนอินเตอร์เพรตเตอร์ต่อไปจนเกือบสมบูรณ์แล้ว

การทดลองใช้งาน เมื่อคอมพายล์แล้วให้รัน MAIN2 จากนี้ให้โหลดโปรแกรมเต่า demo1.lg0 โดยป้อนคำสั่งว่า load demo1.lg0 หลังจากโหลดแล้วให้ป้อนคำสั่งว่า run จะแสดงการทำงานของภาษาล็อกศูนย์ หากคุณต้องการป้อนโปรแกรมเองขอให้ดูถ้อยคำที่ใช้ได้ในโปรแกรม demo1.lg0

แต่ผมก็ขอเรียนให้ทราบก่อนว่าโปรแกรมเหล่านี้มีข้อบกพร่องจะต้องแก้หลายแห่ง แต่ยังไม่มีโอกาสดำเนินการ และผมอาจจะต้องพักเรื่องราวของล็อกศูนย์ไว้หนึ่งฉบับ เพื่อปรับปรุงโปรแกรมก่อนครับ